CNN-Inference on CGRAs under Throughput Constraints

Christian Heidorn, Michael Witterauf, Frank Hannig, and Jürgen Teich
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)
Motivation: CNNs in Autonomous Driving

• Real-time constraints

• Common Accelerators
  • Bigger batch sizes

• Batchsize 1
  • Low latency for one image
  • High throughput (frames per second)
  • GOPS/W

Source: https://electrek.co/2018/10/15/tesla-new-autopilot-neural-net-v9/

Image per Second (Normalized) for mobileNetv1 on Teslas T4 vs Nvidias Volta
Coarse-Grained Reconfigurable Arrays (CGRAs)

- Class of massively parallel processor architectures
  - Array of processing elements (PEs)
  - Stand-alone or accelerator
- Flexibility at runtime
  - Reconfigurable at functional unit level
  - Programmable, reconfigurable interconnect
- Applications
  - Image processing, object recognition
  - Linear algebra (matrix / vector computations)
  - . . . and other streaming applications
I. CNNs
   • Convolutions in CNNs

II. Layer-parallel processing
   • CNN workloads
   • Loop transformation

III. CNNs on CGRAs
   • CGRA example
   • Mapping and calculus
Convolutions in CNNs

- Elementwise multiplication of learned filter parameters (weights) with image pixels
- Accumulation of the partial sums → Output pixel
Convolutions in CNNs

- Many input Feature Maps ($N$)
- Many Filters $\rightarrow$ Many output Feature Maps ($M$)
Convolutional Neural Network (CNN)

<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv0</td>
<td>$M_i \times N_i \times K_{1i} \times K_{2i}$</td>
</tr>
<tr>
<td>Pool1</td>
<td>$K_{p1i} \times K_{p2i}, S_1, S_2$</td>
</tr>
<tr>
<td>Conv2</td>
<td>$M_i \times N_i \times K_{1i} \times K_{2i}$</td>
</tr>
<tr>
<td>Pool3</td>
<td>$K_{p1i} \times K_{p2i}, S_1, S_2$</td>
</tr>
<tr>
<td>Conv4</td>
<td>$M_i \times N_i \times K_{1i} \times K_{2i}$</td>
</tr>
<tr>
<td>Conv5</td>
<td>$M_i \times N_i \times K_{1i} \times K_{2i}$</td>
</tr>
<tr>
<td>Conv6</td>
<td>$M_i \times N_i \times K_{1i} \times K_{2i}$</td>
</tr>
<tr>
<td>Pool7</td>
<td>$K_{p1i} \times K_{p2i}, S_1, S_2$</td>
</tr>
<tr>
<td>FC8</td>
<td>$M_i \times N_i$</td>
</tr>
<tr>
<td>FC9</td>
<td>$M_i \times N_i$</td>
</tr>
<tr>
<td>FC10</td>
<td>$M_i \times N_i$</td>
</tr>
</tbody>
</table>

- Layers are stacked on each other
- Different orders, types, and parameter sizes
I. CNNs
   • Convolutions in CNNs

II. Layer-parallel processing
   • CNN workloads
   • Loop transformation

III. CNNs on CGRAs
   • CGRA example
   • Mapping and calculus
• Increasing number of layers
• Different Workload (Feature map size and weight size) for each layer (operations, layer shapes)
CNNs burdens – AlexNet vs. MobileNet

- Increasing number of layers
- Different Workload (Feature map size and weight size) for each layer (operations, layer shapes)

**Weight profile; Sum 4.67MB**

![Graph showing weight profile for AlexNet](image)

**Weight profile; Sum 6.37 MB**

![Graph showing weight profile for MobileNetV1](image)
• Observation #1: Increasing Feature Map size with more layers appearing in CNNs
• Observation #2: MACs for each layer vary significantly
  → full resource utilization hard to achieve
Layer-by-layer vs. Layer-parallel processing

- **Layer-by-layer execution**

- **Layer-parallel execution**
• Idea: Reduce storage and data transfers for feature maps
• Observation: Each successive layer depends on a small area (denoted as *receptive field*) of the previous one
• I.e., 1 value in $y_{i+1}$ depends on the values in a 4x4 receptive field of the input $x_i$
Storage reduction via layer-parallel execution

• **Layer-by-layer Execution (a)**
  - All input feature maps fully needed to start with computation of next layer

• **Layer-parallel Execution (b)**
  - Only subregion of feature map needed to start computation of next layer
  - Significant reduction of storage (for some nets up to 95%)
  - Reordering of computation needed to achieve a maximal overlap

![Graph a) MobileNetv1, Layer-by-layer](image1)
![Graph b) MobileNetv1, Layer-parallel](image2)
Convolution Layer

Input $X_i$

Output $Y_i$

Parameters $W_i$

$R_{i+1} = \frac{R_i - K1_i}{S_i} + 1$

$N_{i+1} = M_i$

$C_{i+1} = \frac{C_i - K2_i}{S_i} + 1$
Convolution Layer

\[
Y_{i} = \sum_{n} \left( \sum_{k1} \sum_{k2} \right) W[m][n][k1][k2] \times X[n][S*r+k1][S*c+k2] + bias[m];
\]

for (m = 0; m < M; m++)  //number of output feature maps (#filters)
  for (r = 0; r < Rout; r++)  //number of output feature map rows
    for (c = 0; c < Cout; c++)  //number of output feature map columns
      for (n = 0; n < N; n++)  //number of input feature maps
        for (k1 = 0; k1 < K1; k1++)  //filter kernel dimension 1
          for (k2 = 0; k2 < K2; k2++)  //filter kernel dimension 2
            \[ Y[m][r][c] += W[m][n][k1][k2] \times X[n][S*r+k1][S*c+k2] + bias[m]; \]

\[
R_{i+1} = \frac{R_i - K1_i}{S_i} + 1
\]

\[
N_{i+1} = M_i
\]

\[
C_{i+1} = \frac{C_i - K2_i}{S_i} + 1
\]
Loop Transformations

- Loop permutation and unrolling
  - Move loops over filter and no. of feature maps \((n, m)\) to innermost position
  - Allocate multiple filters to a PE, parallel filter execution between PEs
  - Innermost loop \((d)\) corresponds to parallel computation within PEs (ILP)
Convolutional Layer

For $(r = 0; r < R_{out}; r++)$  
for $(c = 0; c < C_{out}; c++)$  
for $(k1 = 0; k1 < K1; k1++)$  
for $(k2 = 0; k2 < K2; k2++)$  
for $(n = 0; n < N; n++)$  
for $(m = p; m < M; m += P)$  
forall $(d = 0; d < \delta; d++)$

\[
Y[m][r][c] += W[m][n+d][k1][k2] \times X[n][S*r+k1][S*c+k2];
\]

if $(k1 == K1 \&\& k2 == K2 \&\& n == N)$
\[
Y[m][r][c] += \text{bias}[m];
\]

\[
R_{i+1} = \frac{R_i - K1_i}{S_i} + 1
\]

\[
N_{i+1} = M_i
\]

\[
C_{i+1} = \frac{C_i - K2_i}{S_i} + 1
\]
I. CNNs
   • Convolutions in CNNs

II. Layer-parallel processing
   • CNN workloads
   • Loop transformation

III. CNNs on CGRAs
   • CGRA example
   • Mapping and calculus
Convolutional Neural Network (CNN)

<table>
<thead>
<tr>
<th>Layer</th>
<th>Shape</th>
<th>Storage (KB) (inp+par+out)</th>
<th>MACs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conv0 (+Pool1)</td>
<td>24 × 1 × 3 × 3</td>
<td>0.8</td>
<td>169,368</td>
</tr>
<tr>
<td>Pool1: 2 × 2, S = 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conv2: 24 × 24 × 3 × 3</td>
<td></td>
<td>19.8 (23.5)</td>
<td>1,016,088</td>
</tr>
<tr>
<td>Pool3: 2 × 2, S = 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conv4: 16 × 24 × 3 × 3</td>
<td></td>
<td>14.5 (5.9)</td>
<td>169,360</td>
</tr>
<tr>
<td>FC5: 10 × 784</td>
<td></td>
<td>5.4</td>
<td>7,850</td>
</tr>
</tbody>
</table>
Mapping of CNN Layers onto 4x4 CGRA

- Conv0
- Conv2
- Conv4
- Pool1
- Pool3
- Weight Buffer
- Input / Output Buffer
Layer 0  Layer 1  Layer 2  Layer 3  Layer 4

\[
\begin{align*}
\text{Conv0} & \quad 4 \text{ PEs} & \quad z_1^{\text{in}} = 216 & \quad z_2^{\text{in}} = 216 & \quad z_3^{\text{in}} = 1,296 & \quad z_4^{\text{in}} = 1,296 \\
& \quad \begin{array}{l}
4 \text{ PEs} \\
\text{Pool1} \\
1 \text{ PE}
\end{array} & \quad z_0^{\text{out}} = 54 & \quad z_1^{\text{out}} = 216 & \quad z_2^{\text{out}} = 48 & \quad z_3^{\text{out}} = \times & \quad z_4^{\text{out}} = \times \\
& \quad \begin{array}{l}
8 \text{ PEs} \\
\text{Conv2}
\end{array} \\
& \quad \begin{array}{l}
1 \text{ PE} \\
\text{Pool3}
\end{array} & \quad z_2^{\text{out}} = 324 & \quad z_3^{\text{out}} = \times & \quad z_4^{\text{out}} = \times & \quad z_4^{\text{out}} = \times
\end{align*}
\]

Required cycles to compute \( M_i \) output pixels:

Number of pixels in \( R_i, C_i \) to start computing:

\[
\begin{align*}
\frac{z_{i}^{\text{out}}}{z_{i}^{\text{in}}} & = \left[ \frac{M_i}{P_i} \right] \cdot \left[ \frac{N_i}{\delta_i} \right] \cdot K1_i \cdot K2_i \\
F_i & = \min(K1_i \cdot K2_i, S1_i \cdot S2_i) \\
\frac{z_{i}^{\text{in}}}{z_{i}^{\text{out}}} & = z_{i-1}^{\text{out}} \cdot F_i \\
z_{i}^{\text{in}} & \leq z_{i}^{\text{out}}
\end{align*}
\]
**Calculus – Layer-Parallel Processing**

### Layer 0 to Layer 4

#### Conv0 (4 PEs)
- \( z_{0}^{\text{in}} = 216 \)
- \( z_{0}^{\text{out}} = 54 \)

#### Pool1 (1 PE)
- \( z_{1}^{\text{in}} = 216 \)
- \( z_{1}^{\text{out}} = 48 \)

#### Conv2 (8 PEs)
- \( z_{2}^{\text{in}} = 216 \)
- \( z_{2}^{\text{out}} = 324 \)

#### Pool3 (1 PE)
- \( z_{3}^{\text{in}} = 1,296 \)
- \( z_{3}^{\text{out}} = 1,296 \)

#### Conv4 (2 PEs)
- \( z_{4}^{\text{in}} = 1,296 \)
- \( z_{4}^{\text{out}} = 864 \)

**Start time layer \( i \):**

\[
t_{i} = \sum_{j=0}^{i} Z_{j} = \sum_{j=0}^{i} z_{j}^{\text{in}}
\]

<table>
<thead>
<tr>
<th>( Z_{i} )</th>
<th>( t_{i} )</th>
<th>( L_{i} )</th>
<th>( L )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>216</td>
<td>42,336</td>
<td>42,336</td>
</tr>
<tr>
<td>0</td>
<td>216</td>
<td>42,336</td>
<td>42,336</td>
</tr>
<tr>
<td>216</td>
<td>432</td>
<td>63,504</td>
<td>63,504</td>
</tr>
<tr>
<td>1,296</td>
<td>1,728</td>
<td>63,504</td>
<td>63,504</td>
</tr>
<tr>
<td>1,296</td>
<td>3,024</td>
<td>63,504</td>
<td>63,504</td>
</tr>
</tbody>
</table>

**Latency layer \( i \):**

\[
L_{i} = z_{i}^{\text{out}} \cdot R_{i+1} \cdot C_{i+1}, \quad L = t_{V-1} + L_{V-1}
\]

**Overall latency:**

\[
L = 66,528
\]
Layer-Parallel Execution Schedule

Layer 0

Input 0

Layer 1

Conv0

Pool1

Layer 2

Conv2

Layer 3

Pool3

Layer 4

Conv4

Output 0

Input 1

Conv0

Pool1

Conv2

Pool3

Conv4

Output 1

\[ L = 66,528 \]
Comparison of 16 PE implementations

Layer-by-Layer       Layer-parallel

Latency $L$ [cycles]  159,936  66,528

Throughput $T$ [fps]   312.6  787.4
Thank you!
Questions?

CNN Inference on CGRAs under Throughput Constraints

Christian Heidorn, Michael Witterauf, Frank Hannig, and Jürgen Teich
References


